

لَهُ مَنْ فِي السَّمَاوَاتِ وَالْأَرْضِ
وَمَا يَرَى إِلَّا مَا أَنْشَأَ
وَإِنَّ رَبَّهُ لَغَنِيمَةٌ
لَهُ مَا فِي الْأَفْوَاتِ
وَمَا يَمْسِي
وَمَا يَمْلِئُ
وَمَا يَمْلِئُ
وَمَا يَمْلِئُ
وَمَا يَمْلِئُ



گزارش پروژه نهایی درس شبکه های تلفن همراه

گزارش

زهرا دهقان

اسماء حمید

فاطمه شرح دهی مقدم

آخرین ویرایش: ۳۱ مرداد ۱۴۰۴ در ساعت ۲۳ و ۵۳ دقیقه

فهرست مطالب

۳	فصل ۱	معرفی پروژه
۴	فصل ۲	Polaris Client
۴		Application Web
۴		نوار کناری (Sidebar)
۵		داشبورد
۶		توضیحات مرتبط با کارت‌ها
۶		دیگر نمودارهای تحلیلی
۱۱		bar Navigation
۱۱		صفحه ورود
۱۱		اکانت ادمین
۱۱		نتیجه‌گیری
۱۲		Android Mobile Client
۱۲		utils
۱۶		SharedViewModel : viewModel
۱۷		MyForegroundService : service
۱۷		تحلیل اکتیویتی‌ها
۱۹		Fragments : UI
۲۳		ProfileInfoItemView : customViews
۲۶	فصل ۳	Polaris Server
۲۶		Authentication
۲۶		Register API (ثبت‌نام کاربر)
۲۸		API Login (ورود کاربر)

۲۹	Logout API (خروج کاربر)	۳.۱.۳
۲۹	Profile API (نمایه کاربر)	۴.۱.۳
۳۰	CellInfo API	۲.۳
۳۰	دسته‌بندی کاربران	۱.۲.۳
۳۰	مدل‌ها	۲.۲.۳
۳۱	مدل‌های اختصاصی هر نسل	۳.۲.۳
۳۲	Serializers	۴.۲.۳
۳۲	Views	۵.۲.۳
۳۴	Tests API	۴.۳
۳۴	ویژگی‌ها	۱.۳.۳
۳۵	مدل‌ها	۲.۳.۳
۳۶	Serializers	۳.۳.۳
۳۶	Views	۴.۳.۳
۳۷	نمونه‌ها (ورودی و خروجی)	۵.۳.۳
۳۹	API Threshold	۴.۳
۳۹	ویژگی‌ها	۱.۴.۳
۳۹	مدل‌ها	۲.۴.۳
۴۰	Serializers	۳.۴.۳
۴۰	Views	۴.۴.۳

۱ معرفی پروژه

این گزارش، به عنوان پروژه نهایی درس شبکه‌های تلفن همراه، در دانشگاه علم و صنعت ایران تهیه شده است. این پروژه شامل یک راهکار جامع و چندبخشی برای جمع‌آوری و تحلیل داده‌های شبکه موبایل است که شامل یک Server-side Component، یک Web Application و یک Android Mobile Client می‌شود.

هدف اصلی این پروژه، توسعه ابزارهایی برای پایش و ارزیابی کیفیت عملکرد شبکه‌های تلفن همراه است. برای این منظور، کلاینت اندروید به صورت مداوم اطلاعات فنی مربوط به شبکه و موقعیت جغرافیایی کاربر را جمع‌آوری کرده و به سرور ارسال می‌کند. این داده‌ها سپس در بک‌اند پردازش شده و برای مشاهده و تحلیل از طریق کلاینت وب در دسترس قرار می‌گیرند.

تمامی کدهای پروژه، از جمله مستندات کامل گزارش، کدهای فرانت‌اند (وب و اندروید) و کدهای بک‌اند، در یک مخزن گیت‌هاب نگهداری می‌شوند. این رویکرد به تضمین شفافیت، قابلیت توسعه و همکاری در آینده کمک می‌کند.

برای مشاهده پروژه کامل و جزئیات فنی آن، می‌توانید به آدرس گیت‌هاب زیر مراجعه کنید:

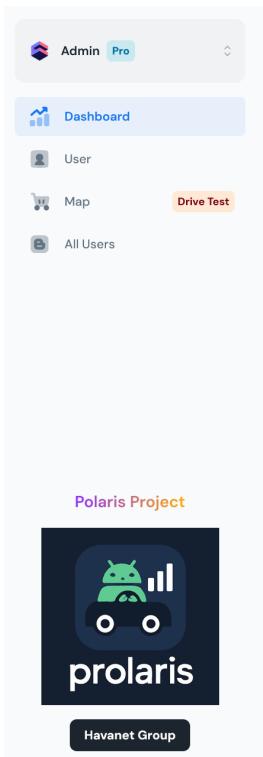
<https://github.com/Havanet-MobileNetWorkProject>

Polaris Client ۲

Application Web ۱.۲

۱.۱.۲ نوار کناری (Sidebar)

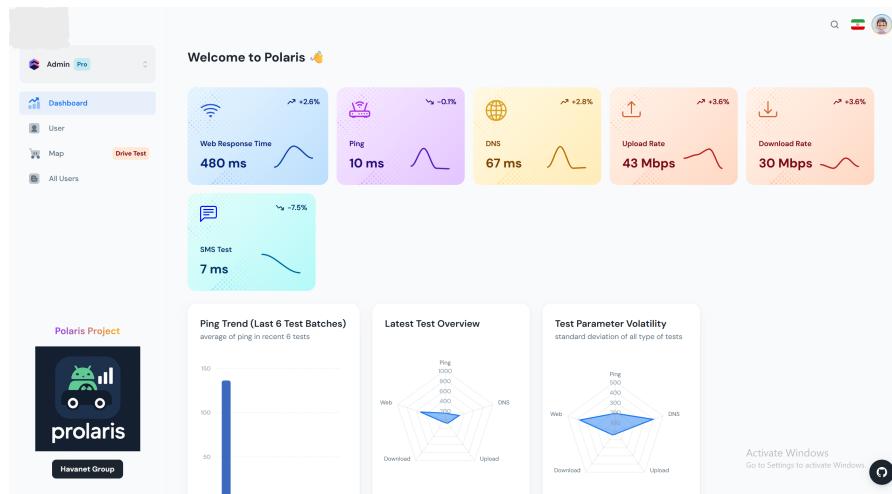
در نوار کناری سامانه Polaris، گزینه هایی وجود دارند که به شما این امکان را می دهند که دسترسی راحتی به امکانات موجود در سامانه داشته باشید. در ادامه به ارائه اطلاعاتی در رابطه با هر ویژگی سامانه می پردازیم.



Sidebar : ۱.۲

۲.۱.۲ داشبورد

در این صفحه، شما دسترسی به تمامی اطلاعات و تجزیه و تحلیل‌های شبکه خواهید داشت. در داشبورد سامانه Polaris شما کارت‌های مختلفی مشاهده خواهید کرد که هر کدام به صورت اختصاصی به نمایش تحلیل نتایج داده تست‌های شما که در اپلیکیشن انجام دادید، می‌پردازند. در ابتدا به توضیح پارامترهایی که مورد تست واقع شده اند می‌پردازیم و سپس نحوه کسب اطلاعات از کارت‌های پارامترها را توضیح می‌دهیم.



شکل ۲.۲: داشبورد

Time Response Web ۱.۲.۱.۲

این کارت نمایانگر میزان تاخیر (Latency) در اتصال به وبسایتها است. در شبکه‌های تلفن همراه، زمان پاسخ‌دهی وب می‌تواند تحت تاثیر عوامل مختلفی مانند بار ترافیکی شبکه، فاصله از سرور و پروتکل‌های ارتباطی باشد. زمانی که این عدد کمتر باشد، تجربه کاربری شما بهینه‌تر خواهد بود و کاهش این زمان باعث بهبود کیفیت تجربه و بگردی می‌شود.

Ping ۲.۲.۱.۲

پینگ یکی از مهم‌ترین پارامترها در ارزیابی کیفیت شبکه تلفن همراه است که نشان‌دهنده تاخیر بین فرستادن درخواست به سرور و دریافت پاسخ از آن است. به‌طور تخصصی، یک پینگ پایین به معنی کیفیت بالای ارتباط در شبکه است. هنگامی که پینگ بالا باشد (مثلاً بالای ۲۰۰ میلی‌ثانیه)، نشان‌دهنده وجود مشکلاتی مانند شلوغی شبکه، محدودیت‌های سرور مقصد یا مشکلات مربوط به پروتکل‌های انتقال داده است.

DNS ۳.۲.۱.۲

زمان DNS معیاری از زمان لازم برای انجام جستجوهای DNS است. اگر شبکه سرعت پایین‌تری در تبدیل نام دامنه به آدرس IP داشته باشد، در این صورت تجربه دسترسی به وبسایتها دچار مشکل می‌شود. در شبکه‌های موبایلی، این زمان می‌تواند تحت

تاثیر کیفیت اتصال شبکه یا بار روی سرورهای DNS قرار گیرد. همچنین در شرایطی که DNS به درستی به روز نشده باشد یا در مسیرهای شبکه موانعی وجود داشته باشد، زمان DNS افزایش خواهد یافت.

Rate Upload/Download ۴.۲.۱.۲

این کارت‌ها سرعت انتقال داده‌ها را در دو جهت بارگذاری و دانلود نمایش می‌دهند. این دو پارامتر نشان‌دهنده ظرفیت شبکه در ارسال و دریافت اطلاعات است. نرخ بالاتر دانلود به معنی سرعت بالای دریافت داده‌ها از اینترنت است، در حالی که نرخ بارگذاری بالا به این معناست که داده‌ها سریعتر به شبکه ارسال می‌شوند.

5.۲.۱.۲ تست پیامک

تست SMS به طور تخصصی برای ارزیابی زمان پیامک‌ها در شبکه‌های تلفن همراه طراحی شده است. در این تست یک پیام با کاربر رد و بدل می‌شود و زمان ارسال پیام تا تایید رسیدن به مقصد به عنوان نتیجه در این کارت نمایش داده می‌شود.

۳.۱.۲ توضیحات مرتبط با کارت‌ها

هر ۶ پارامتر تست شده در اپلیکیشن پولاریس، کارت نمایش نتیجه و تحلیل مخصوص به خودشان را دارند. برای خوانایی بیشتر و تجربه کاربری بهتر تمپلیت هر ۶ کارت یکسان است که به تشریح آن‌ها می‌پردازیم. نتیجه عددی آخرین تست گرفته شده از هر پارامتر در کارت دیده می‌شود. به علاوه نموداری جهت نمایش روند تغییرات n (تست اخیر) میتواند توسط ادمین تغییر کند، در پایین سمت راست هر کارت قابل مشاهده است که با هاور کردن روی آن نتیجه عددی تست‌ها نیز در tooltip به کاربر نمایش داده می‌شود.

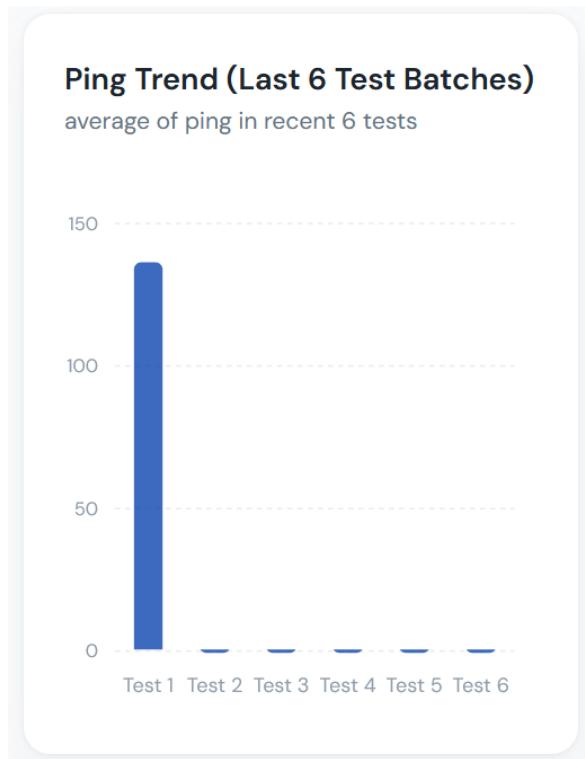
۴.۱.۲ دیگر نمودارهای تحلیلی

1.۴.۱.۲ روند تغییرات پینگ

در این نمودار برای بررسی بهتر تاریخچه تست‌های یک پارامتر و انجام تحلیل دقیق‌تری روی شبکه به دسته بندی که در ادامه شرح میدهم روی آوردیم. هر ۱۲ تست پشت سر هم از نظر زمانی را به عنوان یک دسته یا batch در نظر می‌گیریم. ۶ دسته انتهایی از میان تمامی batch‌ها را انتخاب می‌کنیم. روی هر دسته میانگین گرفته و نتیجه میانگیری را برای هر دسته در نمودار ستونی نمایش می‌دهیم. تست‌های batch اول به نسبت ۶ ام جدیدتر هستند. به طور خلاصه: هر ۶ میله نمودار ستونی نماینده میانگین دوازده تست پشت سر هم از لحاظ زمانی می‌باشد.

2.۴.۱.۲ نمودار Overview Test Latest

این نمودار نمای کلی از وضعیت عملکرد شبکه شما در آخرین تست انجام شده را نمایش می‌دهد. هر پارامتر (مثل، Web, Ping, Upload و Download DNS) در بازه‌ای از ۰ تا ۱۰۰ درصد به نمایش گذاشته می‌شود. در اینجا، هرچه مقدار بیشتر به سمت ۱۰۰ درصد حرکت کند، نشان‌دهنده‌ی کیفیت بهتر آن پارامتر است. برای مثال، اگر Ping نزدیک به ۱۰۰ باشد، یعنی تأخیر شبکه شما



شکل ۳.۲: نتایج تست پینگ

بسیار پایین است، که برای عملکرد خوب شبکه ضروری است. این نمودار به شما کمک می‌کند تا سریعاً مشاهده کنید که کدام پارامترها در شبکه شما عملکرد بهتری دارند و کدام نیاز به بهبود دارند. شاخص کیفیت نیز یک حالت دیفالت دارد که طبق استانداردهای جهانی تعیین شده اما اگر کاربر در اپلیکیشن پولاریس شاخص‌های مد نظر خودش را وارد کند بر طبق همان شاخص‌ها کیفیت پارامتر اندازه‌گیری و در نمودار نمایش داده می‌شود.

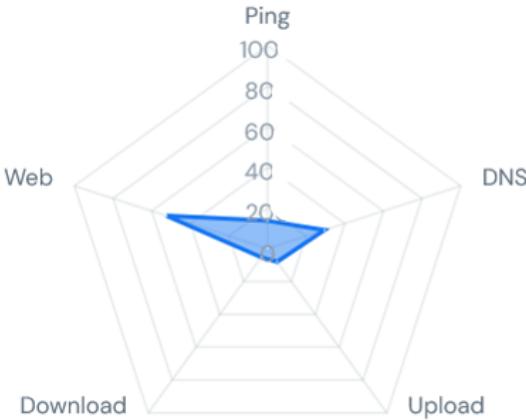
۳.۴.۱.۲ نمودار Volatility Parameter Test

این نمودار نشان‌دهنده‌ی نوسانات یا تغییرات پارامترهای مختلف در طول زمان است. مقیاس این نمودار نیز از ۰ تا ۱۰۰ درصد است و هرچه مقدار بیشتر به سمت ۱۰۰ درصد باشد، نوسانات بیشتری در آن پارامتر مشاهده می‌شود. برای مثال، اگر مقدار Web در این نمودار نزدیک به ۱۰۰ درصد باشد، نشان‌دهنده‌ی نوسانات زیاد در کیفیت وب در طول زمان است، که ممکن است به مشکلات موقتی در شبکه اشاره کند. این ابزار به کاربر کمک می‌کند تا تحلیل دقیقی از ثبات و تغییرات عملکرد شبکه در طی زمان داشته باشد. دوازده تست آخر هر پارامتر در اندازه‌گیری انحراف معیار مد نظر ما بوده است.

۴.۴.۱.۲ Qualities Test Download

این کارت، کیفیت تست‌های دانلود را با استفاده از سه دسته‌بندی، "GoodQ" و "MediumQ" و "LowQ" نمایش می‌دهد. هر بخش نمایانگر درصدی از تست‌ها است که به هر کدام از این سه گروه تعلق دارند. این کارت کمک می‌کند تا کیفیت کلی دانلودها در شبکه خود را مشاهده کنید و ببینید که چه درصدی از تست‌ها به سرعت‌های پایین، متوسط یا خوب تعلق دارند. تعیین شاخص کیفیت بر دو اساس است. اگر کاربر شاخص پارامتری در اپلیکیشن تعریف نکرده باشد، از مقادیر استاندارد جهانی شاخص پارامتر استفاده

Latest Test Overview



شکل ۴.۲: آخرین تست‌ها

میکنیم. در غیر این صورت شاخص کیفیت بر اساس خواسته کاربر در نمودار نمایش داده می‌شود.

Rates Improvement ۵.۴.۱.۲

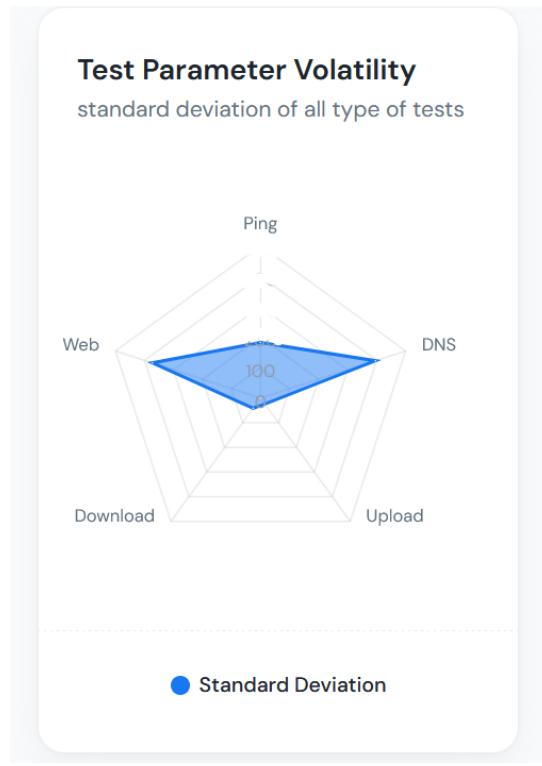
این کارت نشان‌دهنده میزان بهبود در هر یک از پارامترهای تست از جمله، SMS، دانلود، آپلود، DNS و Ping است. به عبارت دیگر، این نمودار به شما نشان می‌دهد که در مقایسه با تست قبلی، عملکرد شبکه در هر یک از این پارامترها به چه میزان تغییر کرده است. این اطلاعات برای شناسایی نوسانات شبکه و تشخیص مشکلات کمک‌کننده هستند. دقت کنید که برای هر پارامتر دو تست آخر در نظر گرفته شده اند که هر کدام درصد بهبود نتیجه خودشان با تست قبلی‌شان را نمایش می‌دهد.

News ۶.۴.۱.۲

کارت خبری شامل اخبار مربوط به صنعت شبکه و تکنولوژی‌ها است. این اخبار می‌توانند شامل اطلاعات جدید درباره بهبود تکنولوژی‌های مختلف، استانداردها یا راهکارهایی برای بهبود عملکرد شبکه باشند. این بخش برای حفظ آگاهی از آخرین تغییرات و نوآوری‌ها در این صنعت مناسب است و لینک اخبار نیز توسط ادمین می‌تواند تغییر کند. دو خبر استاتیک از سایت رسمی gsma به عنوان اخبار دیفالت قرار داده شده.

Site by Traffic ۷.۴.۱.۲

این کارت تعداد کاربران عادی سامانه پولاریس و کاربران با دسترسی ادمین را نمایش می‌دهد. این کاربران همگی در سامانه احراز هویت و ثبت موفق شده اند.



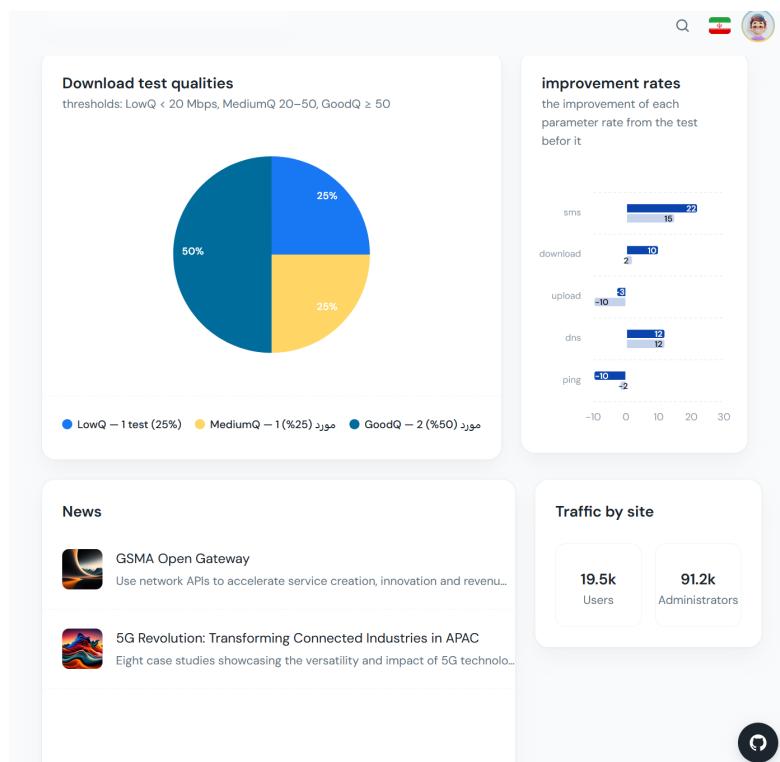
شکل ۵.۲: پارامترهای تست شده

۸.۴.۱.۲ جدول نتایج

برای گزارش بر خط تمامی سیگنال های دریافتی هنگام تست ، این جدول طراحی شده تا تمام ویژگی ها و شاخص های سیگنال نتیجه را گزارش کند. هر ردیف نمایانگر اطلاعات یک رکورد دیتابیس یا همان سیگنال است و هر ستون به یه ویژگی اشاره دارد. قابلیت فیلتر نتایج نیز درنظر گرفته شده تا امکان تحلیل دقیق تر و سریع تر را به کاربر بدهد. در نهایت اگر فیلتر مدنظر شما در گزینه های پیشنهادی نبود باکس سرچی برای شما تعییه کردیم که هر نوع عبارت دلخواه خود که ممکن است در جدول وجود داشته باشد را بیابید. امکان انتخاب یک ردیف یا ردیف های متوالی برای پاک کردن آنها از جدول را نیز دارید. با کلیک بر روی دکمه دانلود اکسل شما میتوانید از کل جدول خروجی اکسل دریافت کنید . با انتخاب ردیف مدنظر خود و کلیک بر دکمه دانلود خروجی kml مدنظرتان را جهت باز کردن در earth google دریافت میکنید. جدول قابلیت اسکرول کردن دارد و باقی ستون ها را میتوانید ببینید. میتوانید جدول خود را با استفاده از علامت فلش رو به بالا یا پایین به ترتیب زمان صعودی یا نزولی مرتب کنید.

Map ۹.۴.۱.۲

در این بخش، شما می توانید اطلاعات موقعیتی مربوط به شبکه را مشاهده کنید. این ابزار به شما امکان می دهد که داده های مربوط به تست های شبکه موبایل و نتایج آن ها را بر اساس موقعیت جغرافیایی روی نقشه نمایش ببینید. برای هر رکورد ثبت شده در درایو تست شما، یک tooltip که شامل اطلاعات آن رکورد است در نظر گرفته شده است.



شکل ۶.۲: ترافیک سایت ، اخبار ، نمودار بھبود و نمودار دایره ای پارامتر

Signal Data

Download CSV Download KML + New Entry

2G, 3G, 4G Choose Technology Choose parameter Clear Filters

1 selected

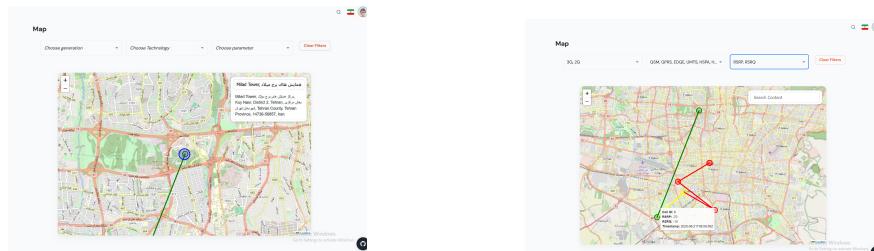
	timestamp ↑	latitude	longitude	cell_id	plmn_id	node_id	tac	lac	band	arfcn	sci
□	8/21/2025, 11:36:37 AM	35.68945	51.3892	67890	12345			21300	1800	100	2G
□	8/21/2025, 12:05:50 PM	35.6893	51.389	513578	43211			32511	5	21300	2G
□	8/21/2025, 12:22:01 PM	35.6695	51.348	1	43211	513578		11	1	120	3G
□	8/21/2025, 12:26:56 PM	35.6586	51.3555	11	43232	513578	32511		3	21300	4G
□	8/21/2025, 4:45:08 PM	35.6395	51.396	1	43232	523642		32511	3	120	3G

Rows per page: 5 ▾ 5 of 5 Activate Windows Go to Settings to activate Windows

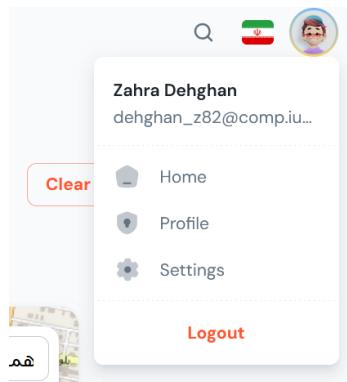
شکل ۷.۲: جدول نتایج

bar Navigation ۵.۱.۲

در نوار بالای صفحه برای تجربه کاربری بهتر قابلیت‌های سرچ، سست کردن پروفایل و انتخاب پرچم کشور کاربر وجود دارد.



شکل ۸.۲: تصاویر نقشه



شکل ۹.۲: Navigation

۶.۱.۲ صفحه ورود

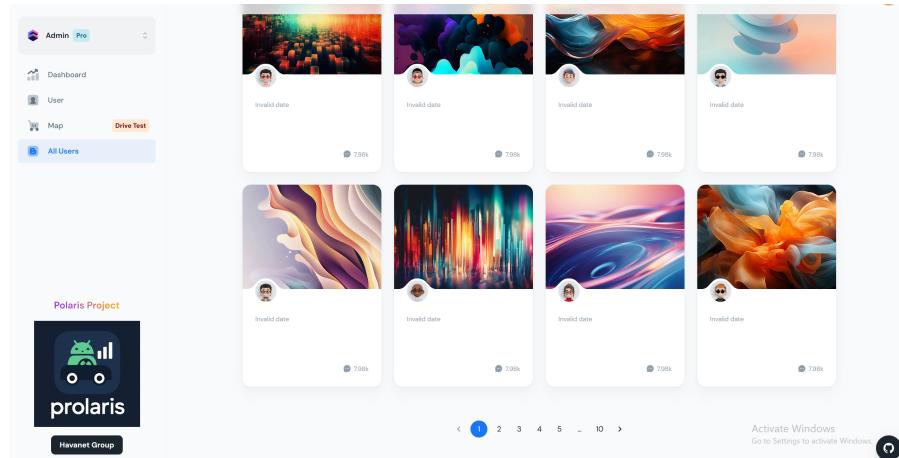
صفحه ورود برای دسترسی امن کاربران به داشبورد خودشان طراحی شده است که صرفاً کاربران ثبت نام کرده در اپلیکیشن اجازه ورود به سامانه با شماره موبایل و رمز مخصوص به خود را دارند.

۷.۱.۲ اکانت ادمین

بزرگترین تفاوت میان اکانت ادمین و عادی این است که کاربر عادی فقط به نتایج تست‌های خودش دسترسی دارد. به عنوان مثال در جدول نتایج سیگنال‌ها، کاربر فقط نتایج تست‌های خودش را می‌بیند در حالی که ادمین می‌تواند همه را ببیند و اگر خواست از قابلیت فیلتر یوزرها استفاده کند.

۸.۱.۲ نتیجه‌گیری

سامانه Polaris ابزاری قدرتمند برای تحلیل و نظارت بر کیفیت شبکه‌های تلفن همراه است. با استفاده از این سامانه می‌توانید وضعیت عملکرد شبکه خود را در زمان‌های مختلف بررسی کرده و مشکلات احتمالی را شناسایی کنید. ابزارهای مختلف تحلیل و گزارش‌دهی در این سامانه به شما کمک می‌کنند تا هر گونه اختلال و نوسانات در شبکه را شناسایی کرده و اقدامات لازم را انجام دهید.



شکل ۱۰.۲: صفحه ادمین

Android Mobile Client ۲.۲

utils ۱.۲.۲

پوشه utils یکی از پایه‌های اصلی معماری تمیز و مازولار در این پروژه است. این فolder شامل مجموعه‌ای از کلاس‌ها و ابزارهای کمکی است که وظایف عمومی و پرکاربرد را از منطق اصلی اپلیکیشن جدا می‌کنند. با مرکز کردن این وظایف در یک مکان مشخص، از تکرار کد جلوگیری می‌شود و خوانایی، قابلیت نگهداری و توسعه‌ی پروژه بهبود می‌یابد.

• مدیریت مجوزهای دسترسی

این کلاس وظیفه مدیریت و بررسی مجوزهای ضروری اپلیکیشن را بر عهده دارد. با استفاده از این کلاس، قبل از انجام هر عملیاتی که نیاز به مجوز دارد (مانند دسترسی به مکان یا ارسال پیامک)، می‌توان از وجود مجوز اطمینان حاصل کرد.

PermissionsUtils.kt : ۱.۲

```

1
2     object PermissionsUtils {
3         val REQUIRED_PERMISSIONS = arrayOf(
4             Manifest.permission.ACCESS_COARSE_LOCATION,
5             Manifest.permission.SEND_SMS,
6             Manifest.permission.RECEIVE_SMS,
7             Manifest.permission.ACCESS_FINE_LOCATION,
8             Manifest.permission.READ_PHONE_STATE
9         )
10
11
12     fun hasAllPermissions(context: Context): Boolean {
13         return REQUIRED_PERMISSIONS.all {
```

```

14     ActivityCompat.checkSelfPermission(
15         context, it) == PackageManager.
16         PERMISSION_GRANTED
17     }
}

```

• ابزار مکان‌یابی

این کلاس یک واسط ساده برای دریافت آخرین موقعیت مکانی کاربر فراهم می‌کند. ابتدا وجود مجوزهای لازم از PermissionsUtils بررسی می‌شود و سپس با استفاده از سرویس‌های Google Play، مکان کاربر به صورت ناهمگام دریافت شده و نتیجه از طریق یک callback برگردانده می‌شود.

کد : ۲.۲ LocationUtils.kt

```

1 object LocationUtils {
2     @SuppressLint("MissingPermission")
3     fun getCurrentLocation(context: Context,
4         onLocationReceived: (Location?) -> Unit) {
5         if (!PermissionsUtils.hasAllPermissions(
6             context)) {
7             onLocationReceived(null)
8             return
9         }
10        val fusedLocationClient = LocationServices.
11            getFusedLocationProviderClient(context)
12        fusedLocationClient.lastLocation
13            .addOnSuccessListener { location: Location? -
14                onLocationReceived(location)
15            }
16            .addOnFailureListener {
17                onLocationReceived(null)
18            }
19        }
}

```

• ابزار اطلاعات شبکه

کلاس NetworkUtils یک ابزار تخصصی برای جمع‌آوری اطلاعات فنی شبکه تلفن همراه است. این کلاس از قابلیت‌های سیستم اندروید برای شناسایی و تحلیل شبکه‌های سلولی اطراف استفاده می‌کند.

- متد getNetworkTypeName

این متد نوع شبکه فعلی دستگاه را به یک رشته قابل فهم تبدیل می‌کند. با استفاده از عبارت when، مقادیر عددی به نام‌های متناظر شبکه (مثل，“HSPA+“，“5G“，“LTE“ وغیره) ترجمه می‌شوند.

کد ۳.۲ : متد getNetworkTypeName

```
1     fun getNetworkTypeName(manager: TelephonyManager
2         ): String {
3             return when (manager.networkType) {
4                 TelephonyManager.NETWORK_TYPE_LTE -> "LTE"
5                 TelephonyManager.NETWORK_TYPE_NR
6                     -> "5G"
7                 TelephonyManager.NETWORK_TYPE_HSPAP -> "HSPA"
8                     +
9                 TelephonyManager.NETWORK_TYPE_HSPA -> "HSPA"
10                TelephonyManager.NETWORK_TYPE_UMTS -> "UMTS"
11                TelephonyManager.NETWORK_TYPE_EDGE -> "EDGE"
12                TelephonyManager.NETWORK_TYPE_GPRS -> "GPRS"
13                TelephonyManager.NETWORK_TYPE_GSM -> "GSM"
14                else -> ""
15            }
16        }
```

- متد getCellInfoText

این متد وظیفه جمع‌آوری اطلاعات کامل از سلول‌های شبکه اطراف را بر عهده دارد:

* دریافت اطلاعات با `.telephonyManager.allCellInfo`

* پردازش داده‌ها برای هر نوع سلول، `:GSM`, `NR`, `LTE`, `WCDMA`,

* استخراج جزئیات فنی مانند `Cell ID`, `PLMN ID`, `Cell ARFCN` و `TAC` و `Signal Quality`.

* سازماندهی داده‌ها در یک شیء `JSONObject` برای ذخیره یا ارسال به سرور.

* بازنگشت خروجی به صورت یک `Pair` شامل پیام متنی و شیء `JSONObject`.

- متدی تبدیل فرکانس (استفاده داخلی)

این متدی برای تبدیل مقادیر `ARFCN/UARFCN/EARFCN/NRARFCN` به فرکانس واقعی در مگاهرتر طراحی

شده‌اند و اطلاعات دقیق‌تری درباره باند فرکانسی ارائه می‌دهند.

پارامتر های دست نیافته

عدم توانایی در دسترسی به مقادیر "RAC" و "Ec/N0" در برنامه های اندرویدی عمدتاً ناشی از محدودیت های نرم افزاری و سخت افزاری سیستم عامل است. این پارامترها، به ویژه Ec/N₀ که مربوط به کیفیت سیگنال شبکه های WCDMA است و RAC که برای شناسایی منطقه مسیریابی در شبکه کاربرد دارد، توسط بسیاری از گوشی ها به صورت مستقیم در اختیار های API عمومی اندروید قرار نمی گیرند. به همین دلیل، در محیط توسعه استاندارد و با استفاده از ابزارهای معمولی مانند Android Studio، امکان استخراج مستقیم این مقادیر محدود و در بسیاری موارد غیرممکن است.



• تست کننده تحويل پیامک

این کلاس یک ابزار دقیق برای تست عملکرد ارسال و دریافت پیامک است. با استفاده از `BroadcastReceiver` زمان تحويل پیامک اندازه گیری شده و تأخیر آن محاسبه می شود.

کد ۴.۲ : SmsDeliveryTester.kt

```

1   class SmsDeliveryTester(
2     private val context: Context,
3     private val onResult: (Long) -> Unit
4   ) {
5     private val deliveryAction = "com.example.Havanet.
6       SMS_DELIVERED"
7     private val deliveryReceiver = object :
8       BroadcastReceiver() {
9         override fun onReceive(ctx: Context?,
10           intent: Intent?) {
11             val sentTime = intent?.getLongExtra
12               ("sentTime", -1L) ?: return
13             val deliveryTime = System.
14               currentTimeMillis()
15             val delay = deliveryTime - sentTime
16             Log.d("SmsDeliveryTester", "SMS
17               delivery delay: $delay ms")
18             context.unregisterReceiver(this)
19             onResult(delay)
20         }
21     }
22
23     fun sendSms() {
24       val sentTime = System.currentTimeMillis()
25
26       val intent = Intent(deliveryAction).apply {
27         putExtra("sentTime", sentTime)}
28     }
29   }
30 }
```

```

۲۲         val pendingIntent = PendingIntent.
۲۳             getBroadcast(
۲۴                 context,
۲۵                 0,
۲۶                 intent,
۲۷                 PendingIntent.FLAG_UPDATE_CURRENT or
۲۸                 PendingIntent.FLAG_IMMUTABLE
۲۹             )
۳۰             context.registerReceiver(deliveryReceiver,
۳۱                 IntentFilter(deliveryAction))
۳۲             val smsManager = SmsManager.getDefault()
۳۳             smsManager.sendTextMessage(phoneNumber,
۳۴                 null, message, null, pendingIntent)
۳۵         }
۳۶     }

```

SharedViewModel : viewModel ۲.۲.۲

کلاس SharedViewModel در پروژه نقشی کلیدی در مدیریت و به اشتراک‌گذاری داده‌ها بین کامپوننت‌های مختلف برنامه ایفا می‌کند. این کلاس با استفاده از ViewModel از کتابخانه Jetpack Android، داده‌ها را از چرخه حیات جدا می‌سازد، به طوری که داده‌ها در زمان تغییر پیکربندی (مانند چرخش صفحه) از بین نمی‌روند.

پیاده‌سازی و عملکرد

- **مدیریت URL پایه:** این کلاس از MutableLiveData برای ذخیره URL پایه سرور (_baseUrl) استفاده می‌کند. این امر به کامپوننت‌های رابط کاربری امکان می‌دهد تا به صورت زنده تغییرات URL را مشاهده کرده و به آن واکنش نشان دهند. متدهای initBaseUrl آدرس IP را به صورت محلی تعریف کرده و آن را در SharedPreferences ذخیره نموده و سپس در _baseUrl قرار می‌دهد.

- **ذخیره‌سازی و بازیابی داده‌ها:** هسته اصلی این ViewModel در متدهای loadData، saveData و clearData قرار دارد که برای مدیریت داده‌های کلید-مقدار (Key-Value) به کار می‌روند. این متدها از SharedPreferences به عنوان یک ابزار سبک وزن برای ذخیره‌سازی داده‌های ساده استفاده می‌کنند.

- **متد saveData:** این متدهای تواند انواع مختلف داده‌ها (Long, Float, Int, Boolean, String) را بر اساس نوع آنها در SharedPreferences ذخیره نماید. این طراحی یک راه حل عمومی و انعطاف‌پذیر برای ذخیره‌سازی داده‌ها فراهم می‌کند.

- **متد loadData:** این متدهای داده‌ها را با یک کلید مشخص و یک مقدار پیش‌فرض از SharedPreferences بازیابی می‌کند. با استفاده از Generics (<T>) و یک عبارت when، این متدهای قادر است داده‌ها را به نوع صحیح خود تبدیل کرده و برگرداند.

- **متد clearData:** امکان حذف یک مقدار خاص از SharedPreferences را با استفاده از کلید آن فراهم می‌کند. این

قابلیت برای مدیریت داده‌های حساس (مانند توکن‌های احراز هویت) که باید پس از خروج کاربر یا در شرایط خاص حذف شوند، بسیار حیاتی است. این متدها با فراخوانی `remove(key)` داده مرتبط را حذف می‌کنند.

- **رویکرد معماری:** استفاده از `ViewModel` به عنوان یک مخزن مرکزی برای داده‌های اشتراکی، از وابستگی مستقیم سایر بخش‌های برنامه به `SharedPreferences` جلوگیری می‌کند. این کار باعث می‌شود که کد تمیزتر، قابل نگهداری تر و تست‌پذیرتر باشد، زیرا منطق ذخیره‌سازی و بازیابی داده‌ها در یک مکان واحد متمرکز شده است.

MyForegroundService : service ۳.۲.۲

به عنوان یک سرویس پس‌زمینه در پروژه، نقش حیاتی در جمع‌آوری و ارسال مداوم اطلاعات فنی به سرور بک‌اند را ایفا می‌کند. این سرویس به صورت *Foreground* طراحی شده تا با نمایش یک نوتیفیکیشن دائمی به کاربر، از بسته شدن ناگهانی آن توسط سیستم عامل اندروید جلوگیری شود. این ویژگی برای عملیات‌های طولانی‌مدت و بدون وقفه ضروری است.

پیاده‌سازی و عملکرد

- **زمان‌بندی هوشمند:** در زمان راه‌اندازی، سرویس یک `Handler` را فعال می‌کند که هر ۱۰ ثانیه یک بار متدهای `collectAndSendData()` را فراخوانی می‌کند. این رویکرد، پایداری و عملکرد بهینه سرویس را تضمین می‌کند.

- **جمع‌آوری داده:** متدهای `collectAndSendData()` و `collectAndSendData()` برای دریافت جزئیات دقیق شبکه (مانند نوع و اطلاعات سلول‌ها) و موقعیت جغرافیایی دستگاه استفاده می‌کند. این طراحی، نشان‌دهنده یک معماری مازولات و تمیز است که وظایف را به ابزارهای تخصصی خود در پکیج `utils` واگذار می‌کند.

- **ارسال به سرور:** پس از جمع‌آوری داده‌ها و ترکیب آن‌ها در یک `JSONObject`، اطلاعات به متدهای `sendToBackend()` و `sendToBackend()` ارسال می‌شود. این ارسال به صورت *Asynchronous* و با استفاده از کتابخانه `OkHttp` انجام می‌شود تا رابط کاربری اصلی (UI) مسدود نشود.

- **مدیریت پیکربندی:** آدرس سرور و توکن احراز هویت به صورت پویا از یک `SharedViewModel` باگذاری می‌شوند که نشان‌دهنده جداسازی داده‌های پیکربندی از منطق اصلی سرویس است.

- **پایش و اشکال‌زدایی:** در صورت موفقیت یا شکست در ارسال داده، پیام‌های مربوطه در `Logcat` ثبت می‌شوند که برای رصد و اشکال‌زدایی عملکرد سرویس بسیار مفید است.

۴.۲.۲ تحلیل اکتیویتی‌ها

MainActivity ۱.۴.۲.۲

به عنوان دروازه ورود به بخش اصلی اپلیکیشن عمل می‌کند و وظایف مهمی مانند مدیریت جریان ورود کاربر، درخواست مجوزهای ضروری و راه‌اندازی سرویس‌های پس‌زمینه را بر عهده دارد.

- مدیریت جریان کاربر: این اکتیویتی اولین نقطه‌ای است که پس از اجرای برنامه بررسی می‌کند که آیا کاربر با موفقیت ثبت‌نام کرده است یا خیر. این کار با بررسی مقدار `isRegistered` در `SharedViewModel` انجام می‌شود. اگر کاربر قبل از احراز هویت نشده باشد، به صورت خودکار به `RegisterActivity` هدایت می‌شود تا فرآیند ورود یا ثبت‌نام را تکمیل کند.
- درخواست مجوزها: در صورت احراز هویت موفق، `MainActivity` مسئولیت درخواست مجوزهای لازم (مانند دسترسی به مکان و شبکه) را به عهده می‌گیرد. این کار از طریق متدهای `onRequestPermissionsResult()` مدیریت می‌شود که پس از دریافت پاسخ کاربر، تصمیم می‌گیرد که برنامه به کار خود ادامه دهد یا بسته شود.
- راه اندازی رابط کاربری و سرویس‌ها: پس از تأیید مجوزها، متدهای `initUI()` رابط کاربری اصلی شامل نوار ابزار و منوی ناوبری پایینی را راه‌اندازی می‌کند. مهم‌تر از آن، در همین مرحله سرویس `MyForegroundService` برای شروع جمع‌آوری داده‌ها به صورت پس‌زمینه فعال می‌شود. این رویکرد جداسازی کامل رابط کاربری از منطق کسب‌وکار را نشان می‌دهد.



: Layout Resource File مسیر

« app\src\main\res\layout\activity_main.xml »

RegisterActivity ۲.۴.۲.۲

یک واسط کاربری متمرکز برای فرآیندهای حساس ثبت‌نام و ورود به سیستم است و تمام ارتباطات لازم با سرور احراز هویت را مدیریت می‌کند.

- واسط کاربری پویا: این اکتیویتی دارای یک حالت دوقطبی است که به کاربر اجازه می‌دهد بین «ورود» و «ثبت‌نام» جابه‌جا شود. این تغییر حالت، عنوان صفحه و متن دکمه‌ها را به صورت پویا تغییر می‌دهد تا تجربه کاربری بهتری فراهم شود.
- ارتباط امن با سرور: متدهای `sendToBackend()` از کتابخانه `OkHttp` برای ارسال اطلاعات احراز هویت به صورت امن استفاده می‌کند. این متدهای اساس وضعیت فعلی (ورود یا ثبت‌نام)، درخواست را به یکی از دو نقطه پایانی `/auth/login/` یا `/auth/signup/` ارسال می‌کند.
- مدیریت پاسخ سرور: پس از دریافت پاسخ موفق، توکن‌های `access` و `refresh` از پاسخ JSON استخراج و با استفاده از `SharedPreferences` در `SharedViewModel` ذخیره می‌شوند. این توکن‌ها برای درخواست‌های بعدی به سرور استفاده خواهند شد. سپس برنامه کاربر را به `MainActivity` هدایت می‌کند و اکتیویتی فعلی را می‌بندد تا از دسترسی غیرمجاز به اطلاعات جلوگیری شود.
- مدیریت خطای خطا: در صورت بروز خطای سمت سرور (مانند نام کاربری یا رمز عبور نامعتبر)، یک پیام خطای واضح به صورت `Toast` به کاربر نمایش داده می‌شود تا از تجربه ناموفق خود آگاه شود.



: Layout Resource File مسیر

« app\src\main\res\layout\activity_register.xml »

ProfileActivity ۳.۴.۲.۲

به کاربر امکان مشاهده جزئیات پروفایل و کنترل جلسه خود را می‌دهد. این اکتیویتی نشان‌دهنده نحوه استفاده از توکن‌های احراز هویت برای دسترسی به اطلاعات حفاظت‌شده است. با کلیک بر علامت آدمک در خط بالای صفحه اصلی میتوان وارد این بخش شد.

- **دریافت اطلاعات پروفایل:** متدها fetchProfile() مسئولیت دریافت اطلاعات کاربر را بر عهده دارد. این متدها با استفاده از token access ذخیره‌شده، یک درخواست GET به نقطه پایانی /auth/profile ارسال می‌کند. این رویکرد تضمین می‌کند که فقط کاربران احراز هویت‌شده به اطلاعات خود دسترسی داشته باشند.
- **نمایش اطلاعات:** اطلاعات دریافتی (مانند شماره تلفن و نقش کاربری) در کامپوننت‌های سفارشی ProfileInfoItemView نمایش داده می‌شوند. این استفاده از ویوهای سفارشی، به یکپارچگی و طراحی منسجم رابط کاربری کمک می‌کند.
- **خروج امن:** متدها performLogout() فرآیند خروج امن را مدیریت می‌کند. این متدها با ارسال refresh token به نقطه پایانی /auth/logout، توکن‌ها را در سرور باطل می‌کنند. سپس با استفاده از SharedViewModel، تمام توکن‌ها و وضعیت ثبت‌نام (isRegistered) را از حافظه محلی دستگاه حذف می‌کنند.
- **قطع سرویس پس‌زمینه:** پس از خروج موفق، MyForegroundService متوقف می‌شود تا جمع‌آوری داده‌ها به پایان برسد. سپس کاربر به صفحه RegisterActivity هدایت می‌شود و فرآیند ورود مجدد آغاز می‌گردد. این کار امنیت داده‌ها و حریم خصوصی کاربر را تضمین می‌کند.

: Layout Resource File



« app\src\main\res\layout\activity_profile.xml »

Fragments : UI ۵.۲.۲

در این بخش، به جزئیات هر فرگمنت که در MainActivity نمایش داده می‌شود، پرداخته می‌شود.

InformationFragment ۱.۵.۲.۲

یکی از مهم‌ترین بخش‌های رابط کاربری اپلیکیشن است که وظیفه نمایش اطلاعات لحظه‌ای مربوط به موقعیت مکانی و جزئیات فنی شبکه تلفن همراه را بر عهده دارد. این فرگمنت با استفاده از معماری MVVM (Model-View-ViewModel) پیاده‌سازی شده و داده‌ها را به صورت مستقیم از لایه‌های viewmodels و utils دریافت می‌کند.

پیاده‌سازی و عملکرد

- **اتصال به ViewModels:** این فرگمنت به دو ViewModel متصل می‌شود:

• **یک ViewModel محلی که برای مدیریت داده‌های مربوط به خود فرگمنت (مانند متن دکمه) استفاده می‌شود.**

• **یک ViewModel مشترک که برای بررسی وضعیت احراز هویت کاربر (isRegistered) و دسترسی به داده‌های عمومی مانند Base URL استفاده می‌شود.**

- **بروزرسانی لحظه‌ای:** برای نمایش اطلاعات لحظه‌ای، از یک Runnable و Handler استفاده شده است. متدهای زیر هر یک ثانیه اجرا می‌شود تا اطلاعات جدید مکان و شبکه را جمع‌آوری و روی UI نمایش دهد.

```
1     private fun updateLocationAndNetworkInfo() {
2         LocationUtils.getCurrentLocation(requireContext())
3             { location ->
4                 if (location != null) {
5                     latitude = location.latitude
6                     longitude = location.longitude
7                     binding.latitudeText.text = " ${latitude} "
8                     binding.longitudeText.text = " ${longitude} "
9                 } else {
10                     binding.latitudeText.text = "- -"
11                     binding.longitudeText.text = "- -"
12                 }
13             }
14             binding.cellinfoTable.removeAllViews()
15             val (_, cellJson) = NetworkUtils.getCellInfoText(
16                 requireContext()
17             )
18             addCellInfoToTable(cellJson)
19         }
20     }
```

• **مدیریت وضعیت کاربر:** قبل از نمایش اطلاعات، فرگمنت وضعیت isRegistered را برسی می‌کند. اگر کاربر احراز هویت نشده باشد، کارت‌های نمایش اطلاعات (dataCard و locationCard) مخفی شده و یک پیام خطاب نمایش داده می‌شود.

• **نمایش اطلاعات:**

- **موقعیت مکانی:** مختصات جغرافیایی (longitude, latitude) با استفاده از LocationUtils.getCurrentLocation() دریافت و نمایش داده می‌شود.

- **اطلاعات شبکه:** جزئیات فنی شبکه با استفاده از NetworkUtils.getCellInfoText() دریافت شده و به صورت TableLayout پویا در یک قرار می‌گیرند.

• **تعامل با نقشه‌ها:** دکمه Open in Map (OSM) ایجاد می‌کند تا مختصات روی نقشه (Intent) نمایش داده شود.

نمایش داده‌های پویا در جدول

• **تولید جدول پویا:** متدهای addCellInfoToTable() برای هر زوج کلید-مقدار یک سطر جدید در TableLayout ایجاد می‌کند.

- قالب‌بندی خوانا: متدهای `createTableRow()` و `createTableHeader()` برای ایجاد سطر و سرمهای جدول است. می‌دهد تا اطلاعات به صورت شفاف نمایش داده شوند.

مدیریت چرخه حیات فرگمنت

- ساخت View فرگمنت و شروع بروزرسانی‌های دوره‌ای در `onCreateView()`.
- حذف `Handler` از `Runnable` برای جلوگیری از Memory Leak و تنظیم `null = _binding` برای `onDestroyView()`.
- کمک به Garbage Collection.

: Layout Resource File



« app\src\main\res\layout\fragment_information.xml »

SettingFragment ۲.۵.۲.۲

یک فرگمنت کلیدی در رابط کاربری اپلیکیشن است که به کاربر امکان تعریف آستانه‌ها و مقادیر رنگی برای نمایش کیفیت سیگنال شبکه‌های مختلف را می‌دهد. این فرگمنت با فراهم کردن یک واسط کاربری پویا، امکان شخصی‌سازی نمایش داده‌های فنی را به صورت بصری فراهم می‌سازد.

پیاده‌سازی و عملکرد

- اسپینرها (Spinners): این فرگمنت دارای سه وابسته به هم است:
 - spinnerTech: انتخاب نوع شبکه، ۴G، ۳G، ۲G، ۵G.
 - spinnerType: بر اساس انتخاب تکنولوژی، گزینه‌های مربوط به نوع سیگنال (مانند rsrq یا rsrp برای ۴G) را بارگذاری می‌کند.
 - spinnerNumber: انتخاب تعداد سطوح رنگی (از ۳ تا ۵۰)، که به صورت مستقیم بر تعداد باکس‌های رنگی قابل تعریف تأثیر دارد.
- انتخاب رنگ: با استفاده از انتخابگر رنگ، رنگ انتخاب شده در متغیر `currentColor` ذخیره می‌شود. کاربر می‌تواند با کلیک روی دکمه مربوطه، دیالوگ `showLevelInputDialog()` را برای ثبت جزئیات سطح رنگ فراخوانی کند.
- دیالوگ ورودی سطح: متدهای `showLevelInputDialog()` و `showColorPickerDialog()` یک دیالوگ سفارشی نمایش می‌دهد که شامل ورودی‌های سطح، حداقل و حداکثر است. این متدهای امکان ویرایش مقادیر سطح‌های قبلی را نیز فراهم می‌کند.
- بهروزرسانی رابط کاربری: متدهای `updateColorViews()` برای هر سطح، یک ویو جدید ایجاد کرده و رنگ و سطح آن را نمایش می‌دهد. همچنین برای هر ویو یک `OnClickListener` تعریف می‌شود تا کاربر بتواند در صورت نیاز وارد حالت ویرایش شود.

ارتباط با سرور و منطق تجاری

- ثبت نهایی: دکمه ثبت نهایی توسط متدهای `createAndSetupFinalizeButton()` ایجاد می‌شود. این متدهای پیش از ارسال، داده‌های کاربر را اعتبارسنجی کرده و در قالب JSON آماده می‌کند.
- ارسال داده‌ها: متدهای `Final_send()` مسئول ارسال داده‌ها به سرور است. این متدها با استفاده از `SharedViewModel`، توکن `/thresholds/create` را دریافت کرده و از کتابخانه `OkHttp` برای ارسال درخواست `POST` به نقطه پایانی `Base URL` احراز هویت و `OkHttp` استفاده می‌کند.
- بازخورد به کاربر: در صورت موفقیت، یک پیام `Toast` نمایش داده شده و رابط کاربری ریست می‌شود.

مدیریت چرخه حیات فرگمنت

- ایجاد رابط کاربری و انجام تنظیمات اولیه. `onCreateView()`
- آزادسازی منابع و تنظیم `_binding = null` برای جلوگیری از `Memory Leak`. `onDestroyView()`



: Layout Resource File

« app\src\main\res\layout\fragment_setting.xml »

TestsFragment ۳.۵.۲.۲

TestsFragment یکی از فرگمنت‌های اصلی اپلیکیشن است که به کاربر امکان اجرای تست‌های عملکردی شبکه و سرویس‌های موبایل را می‌دهد. این فرگمنت با فراهم کردن یک واسطه کاربری ساده، تست‌های مختلفی مانند پینگ، سرعت آپلود/دانلود و تأخیر پیامک را به صورت خودکار و زمان‌بندی شده انجام می‌دهد.

پیاده‌سازی و عملکرد

- رابط کاربری تعاملی: این فرگمنت دارای دکمه‌هایی برای هر تست (پینگ، وب، آپلود، دانلود، DNS و SMS) است. با کلیک روی هر دکمه، تست مربوطه آغاز می‌شود.
- مدیریت وضعیت دکمه‌ها: متدهای `setAllButtonsEnabled()` و `setTestButtonsEnabled()` تمام دکمه‌ها را مدیریت می‌کند. هنگام اجرای یک تست، تمام دکمه‌ها غیرفعال می‌شوند تا از اجرای همزمان چند تست جلوگیری شود.
- بروزرسانی زنده رابط کاربری: با استفاده از `testsViewModel` و `LiveData<TestResult>`، نتایج تست‌ها به صورت لحظه‌ای روی UI نمایش داده می‌شود. این طراحی باعث می‌شود که UI حتی در تست‌های طولانی نیز پاسخگو باقی بماند.

اجرای تست‌ها و منطق مربوطه

- متدهای `startRepeatedTest()` و `stopRepeatedTest()`: هسته اصلی اجرای تست‌ها است.

- زمان‌بندی: با استفاده از `lifecycleScope` و `Coroutines`، تست انتخاب شده را به مدت ۲ دقیقه و با فاصله زمانی ۱ ثانیه تکرار می‌کند.

- همگامی با UI: عملیات شبکه در `IO`. `Dispatchers.Main` به UI منتقل می‌شوند تا از مسدود شدن رابط کاربری جلوگیری شود.

• تست‌های مختلف:

- ارسال پینگ به آدرس ۸.۸.۸.۸ و محاسبه زمان پاسخ: `performPingTest()`

- `https://www.google.com/search?q=google.com`: محاسبه زمان اتصال به `testWebResponseTime()` - `testUploadSpeed()`: اندازه‌گیری سرعت آپلود داده به سرور تستی.

- `testDownloadSpeed()`: اندازه‌گیری سرعت دانلود از سرور تستی.

- `www.google.com`: محاسبه زمان پاسخ `testDnsTime()` -

- `SmsDeliveryTester`: محاسبه زمان تأخیر ارسال/دریافت پیامک با استفاده از `testSmsDeliveryDelay()` و مدیریت ناهمگام با `Coroutines`.

مدیریت داده و ارتباط با سرور

• ارتباط با `SharedViewModel`: برای دسترسی به `Access Token` و `Base URL` استفاده می‌شود تا مدیریت داده‌های مشترک متتمرکز باشد.

• ارسال نتایج به بک‌اند: متد `sendTestResultToBackend()` نتایج تست‌ها را در قالب درخواست `POST` و با فرمت `JSON` به سرور ارسال می‌کند. احراز هویت از طریق `Token Bearer` انجام می‌شود. این کار امکان ذخیره‌سازی و تحلیل نتایج در سمت سرور را فراهم می‌کند.

• مدیریت چرخه حیات: در متد `onDestroyView()`، ارجاع `binding` برابر `null` قرار داده می‌شود تا از `Memory Leak` جلوگیری گردد.

: Layout Resource File



« app\src\main\res\layout\fragment_tests.xml »

ProfileInfoItemView : customViews ۶.۲.۲

یک کامپوننت UI سفارشی است که به منظور نمایش یکپارچه اطلاعات کاربر (مانند نام کاربری، رمز عبور یا نقش) در صفحه `ProfileActivity` طراحی شده است. این رویکرد به کدنویسی تمیزتر و قابل نگهداری کمک می‌کند، زیرا از تکرار کد مربوط به طراحی ویوهای مشابه جلوگیری می‌کند. همچنین به این روش، در صورت نیاز، گسترش این بخش ساده‌تر می‌شود.

۱. پیاده‌سازی و ساختار

- کلاس: این ویو از کلاس پایه `LinearLayout` ارث بری می‌کند و می‌تواند عناصر داخلی خود را به صورت خطی (افقی یا عمودی) سازماندهی کند.
- اتصال به طرح‌بندی: در بلوک `init`, فایل XML مربوط به طرح‌بندی (`layout.profile_info_item`) با استفاده از `LayoutInflator` به ویو متصل می‌شود.
- دریافت ویژگی‌های XML: با استفاده از `context.obtainStyledAttributes`, ویو می‌تواند ویژگی‌های سفارشی تعریف شده در فایل XML مانند `app:text` یا `app:icon` را بخواند و ظاهر ویو را از طریق XML سفارشی‌سازی کند.

۲. عملکرد و قابلیت‌های کلیدی

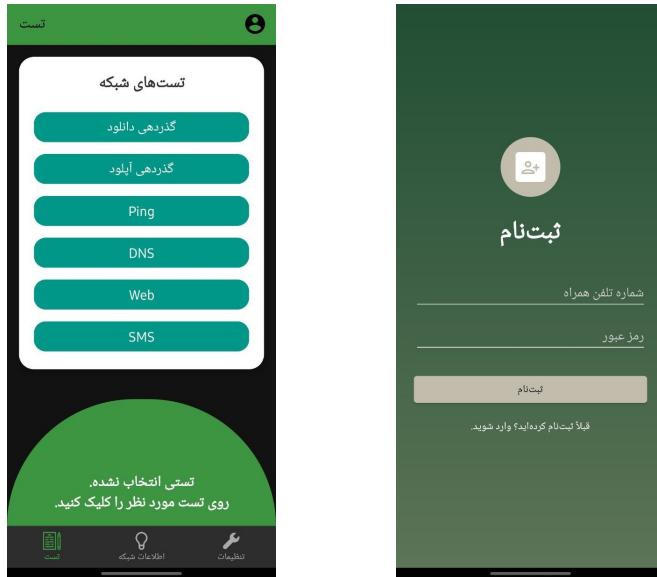
قابلیت استفاده مجدد: توسعه‌دهنده می‌تواند به جای تکرار کد، از تگ `<com.example.Havanet.customviews.ProfileInfoItemView>` در XML استفاده کند.

- کپسوله‌سازی منطق: منطق مربوط به نمایش آیکون و متن داخل کلاس قرار دارد و کد `ProfileActivity` تمیزتر می‌شود.
- متدهای عمومی:
 - `String setValue(value: String)`: تنظیم متن ویو از طریق کد.
 - `String getValue()`: دریافت متن فعلی ویو برای ذخیره یا استفاده از اطلاعات.



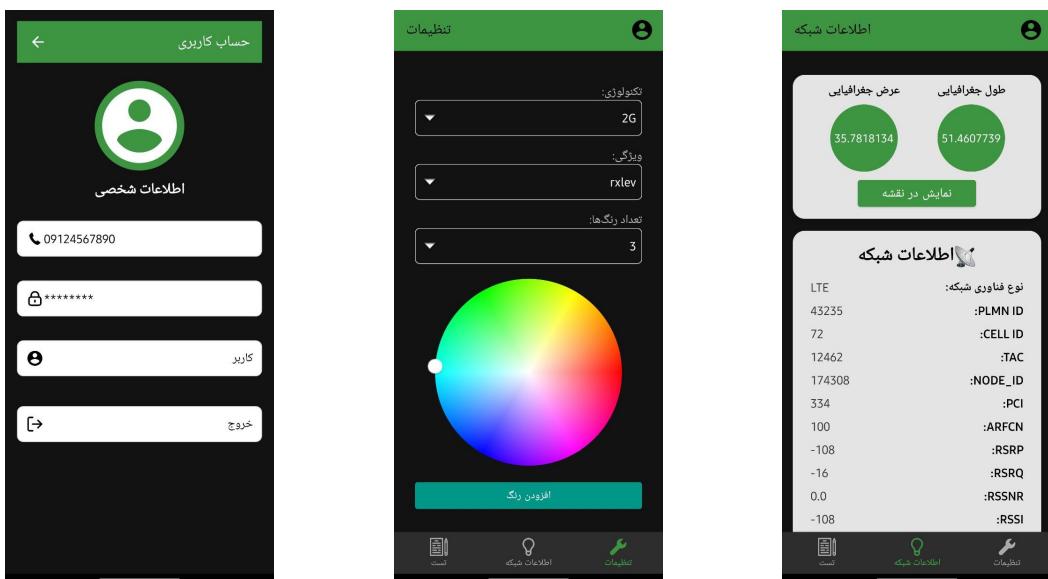
: Layout Resource File مسیر

« app\src\main\res\layout\profile_info_item.xml »



(ب) تست ها

(آ) ورود/ ثبت نام



(ه) حساب کاربری

(د) تنظیمات

(ج) اطلاعات

شکل ۱۱.۲: تصاویر صفحات نام برده شده

Authentication ۱.۳

۱.۱.۳ Register API (ثبت نام کاربر)

(User Model)

مدل کاربر برای سامانه Polaris به شکل زیر طراحی شده است:

- phone: شماره تلفن کاربر (یونیک و بعنوان نام کاربری استفاده می‌شود).
- role: نقش کاربر که می‌تواند user یا admin باشد (پیش‌فرض: user).
- is_active: تعیین فعال بودن کاربر.
- is_staff: برای تشخیص دسترسی مدیریتی.

کاربران عادی نقش user دارند و مدیران (superuser) نقش .admin

همچنین یک UserManager پیاده‌سازی شده تا:

- تابع create_user: کاربر عادی بسازد.
- تابع create_superuser: ادمین بسازد.

(RegisterSerializer) Serializer

کلاس RegisterSerializer برای اعتبارسنجی داده‌های ورودی ثبت‌نام استفاده می‌شود:

- فقط دو فیلد دریافت می‌کند: phone و password.
- password فقط نوشتنی (write_only) است.
- تابع از متدهای create_user در UserManager استفاده می‌کند.

(RegisterView) View

کلاس RegisterView یک APIView است که درخواست‌های POST را پردازش می‌کند:

- داده‌های ورودی توسط RegisterSerializer بررسی می‌شود.

- اگر معتبر بود: کاربر ساخته می شود و توکن های JWT صادر می گردد.
- خروجی شامل توکن ها + اطلاعات کاربر است.

: (POST /auth/signup/)

```

1 {
2   "phone": "09123456789",
3   "password": "test1234"
4 }
```

خروجی موفق (201):

```

1 {
2   "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1...",
3   "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1...",
4   "user": {
5     "id": 1,
6     "phone": "09123456789",
7     "role": "user"
8   }
9 }
```

خروجی ناموفق (400):

```

1 {
2   "phone": ["This field must be unique."]
3 }
```

1.1.3 Signal Post-Save (ایجاد آستانه های پیش فرض بعد از ثبت نام)

پس از ثبت نام هر کاربر جدید، یک سیگنال post_save در جنگو اجرا می شود. این سیگنال باعث می شود که به طور خودکار مجموعه ای از ThresholdLevel و ThresholdParameter برای کاربر ایجاد گردد.

کلیت Threshold

جزئیات کامل ساختار و کاربرد Threshold ها در فصل مربوط به API thresholds توضیح داده خواهد شد. در این بخش تنها نحوه ایجاد پیش فرض ها هنگام ثبت نام بیان می شود.

(create_default_thresholds) Signal

این تابع سیگنال پس از ذخیره هی یک کاربر (User) اجرا می شود:

- بررسی می کند که آیا کاربر تازه ساخته شده است و نقش او user باشد.
- اگر درست باشد، بر اساس default_thresholds برای هر فناوری (2G, 3G, 4G, 5G) پارامترها ساخته می شوند.

- برای هر پارامتر چندین سطح (ThresholdLevel) با بازه مقادیر و رنگ مشخص ایجاد می‌شود.

مقادیر پیشفرض (default_thresholds)

این مقادیر برای هر فناوری از قبل تعریف شده‌اند.

نمونه برای :2G

```

1 "2G": [
2   {
3     "name": "rxlev",
4     "signal_type": "quantity",
5     "levels": [
6       {"level": 1, "color": "#FF0000", "min": -110, "max": -100},
7       {"level": 2, "color": "#FFFF00", "min": -100, "max": -80},
8       {"level": 3, "color": "#008000", "min": -80, "max": -50}
9     ]
10   }
11 ]

```

برای سایر فناوری‌ها (3G، 4G، 5G) نیز پارامترهایی مانند rsrp، rsrq و rscp تعریف شده‌اند که هرکدام دارای سطوح رنگی مختلف هستند.

به این ترتیب، هر کاربر جدید بلافضله پس از ثبت‌نام، یک مجموعه‌ی اولیه از آستانه‌های سنجش کیفیت و کمیت سیگنال را در اختیار خواهد داشت.

۲.۱.۳ API Login (ورود کاربر)

(LoginView) View

- درخواست POST می‌گیرد.
- شماره تلفن و رمز عبور را بررسی می‌کند.
- در صورت معتبر بودن، توکن‌های JWT صادر می‌کند.
- اطلاعات کاربر بازگردانده می‌شود.

ورودی :(POST /auth/login/)

```

1 {
2   "phone": "09123456789",
3   "password": "test1234"
4 }

```

خروجی موفق (200) :

```
1 {
2   "refresh": "eyJ0eXAiOiJKV1QiLCJh...",
3   "access": "eyJ0eXAiOiJKV1QiLCJh...",
4   "user": {
5     "id": 1,
6     "phone": "09123456789",
7     "role": "user"
8   }
9 }
```

خروجی ناموفق (401):

```
1 {
2   "error": "Invalid credentials"
3 }
```

Logout API ۳.۱.۳ (خروج کاربر)

(LogoutView) View

- درخواست POST دریافت می‌کند.
- کاربر باید توکن refresh را ارسال کند.
- توکن بلاکلیست می‌شود.

: (POST /auth/logout/)

```
1 {
2   "refresh": "eyJ0eXAiOiJKV1QiLCJh..."
3 }
```

خروجی موفق (205): (بدون بدنه)

خروجی ناموفق (400): (بدون بدنه)

Profile API ۴.۱.۳ (نمایه کاربر)

(UserSerializer) Serializer

برای نمایش اطلاعات کاربر از UserSerializer استفاده می‌شود (شامل id، phone، role).

(ProfileView) View

- درخواست GET دریافت می‌کند.
- نیاز به احراز هویت دارد.
- اطلاعات کاربر لاغین شده بازگردانده می‌شود.

ورودی (GET /auth/profile/):

خروجی موفق (200):

```
1 {  
2   "id": 1,  
3   "phone": "09123456789",  
4   "role": "user"  
5 }
```

خروجی ناموفق (401):

```
1 {  
2   "detail": "Authentication credentials were not provided."  
3 }
```

CellInfo API ۲.۳

برای ثبت و دریافت اطلاعات سیگنال سلولی کاربران طراحی شده است. این API انواع شبکه‌ها را پشتیبانی می‌کند: 2G، 3G، 4G، 5G. کاربران باید احراز هویت شده باشند (IsAuthenticated) تا بتوانند داده‌ها را ارسال یا دریافت کنند.

۱.۲.۳ دسته‌بندی کاربران

- کاربران عادی: فقط داده‌های خودشان را می‌بینند.
- ادمین‌ها: می‌توانند داده همه کاربران را مشاهده کنند و در صورت نیاز با پارامتر client_id فیلتر کنند.

۲.۲.۳ مدل‌ها

تمام مدل‌ها از BaseSignalTest ارث بری می‌کنند.

BaseSignalTest ۱.۲.۲.۳

- زمان ثبت سیگنال (DateTimeField) timestamp
- نوع تکنولوژی سلولی (CharField) technology
 - 'HSPA+', 'HSPA', 'UMTS', 'EDGE', 'GPRS', 'GSM'، گزینه‌ها
 - '5G', 'LTE-Adv', 'LTE'
- موقعيت جغرافيايی سیگنال (FloatField) longitude و latitude
- شناسه شبکه تلفن همراه (CharField) plmn_id
- شناسه سلول (BigIntegerField) cell_id
- فرکانس راديوسي (IntegerField, optional) arfcn
- optional: باند فرکانسي (IntegerField, ← band)

Generation Property ۲.۲.۲.۳

- GSM, GPRS, EDGE :2G
- UMTS, HSPA, HSPA+ :3G
- LTE, LTE-Adv :4G
- 5G :5G
- سایر مقادیر: Unknown

۳.۲.۳ مدل‌های اختصاصی هر نسل

SignalTest2G ۱.۳.۲.۳

- ارثبری از BaseSignalTest
- کاربر ثبت‌کننده user (ForeignKey → User)
- کد منطقه سلولی lac (IntegerField)
- قدرت سیگنال rxlev (FloatField, optional)

SignalTest3G ۲.۳.۲.۳

- کاربر ثبت‌کننده user (ForeignKey → User)
- کد منطقه سلولی lac (IntegerField)
- قدرت سیگنال راديوسي rscp (FloatField)
- شناسه node_id (BigIntegerField)

SignalTest4G ۳.۳.۲.۳

- user (ForeignKey → User) •
- LTE : tac (IntegerField)
- LTE : rsrp (FloatField)
- LTE : rsrq (FloatField, optional)
- node_id (BigIntegerField)

SignalTest5G ۴.۳.۲.۳

مشابه 4G شامل node_id, rsrq, rsrp, tac

Serializers ۴.۲.۳

SignalTest5GSerializer SignalTest4GSerializer, SignalTest3GSerializer, SignalTest2GSerializer, •

- نقش: تبدیل داده‌های مدل به JSON و اعتبارسنجی داده‌های ورودی
- فیلد user فقط خواندنی است و هنگام POST از کاربر جاری پر می‌شود
- تمام فیلدات دیگر بر اساس مدل هستند ('.fields = '_all_')

Views ۵.۲.۳

(APIView) UnifiedSignalTestView ۱.۵.۲.۳

POST /cellinfo/signal/

- عملکرد: ثبت یک رکورد سیگنال جدید
- احراز هویت: الزامی (IsAuthenticated)
- ورودی: JSON شامل فیلدات مدل مرتبط با تکنولوژی

توضیحات کد POST

۱. دریافت داده‌ها از کاربر

۲. انتخاب Serializer مناسب بر اساس تکنولوژی

۳. اعتبارسنجی داده‌ها توسط Serializer

۴. ذخیره داده‌ها در دیتابیس و اختصاص user به کاربر جاری

۵. برگرداندن Response موفق (201 Created) یا خطأ (400 Bad Request)

مثال ورودی 4G POST \cellinfo \signal

```

1 {
2   "technology": "LTE",
3   "timestamp": "2025-08-21T12:00:00Z",
4   "latitude": 52.37,
5   "longitude": 4.89,
6   "plmn_id": "12345",
7   "cell_id": 67890,
8   "tac": 101,
9   "rsrp": -85.0,
10  "rsrq": -10.0,
11  "node_id": 5555
12 }

```

Response Success 201

```

1 {
2   "id": 1,
3   "user": 2,
4   "technology": "LTE",
5   "timestamp": "2025-08-21T12:00:00Z",
6   "latitude": 52.37,
7   "longitude": 4.89,
8   "plmn_id": "12345",
9   "cell_id": 67890,
10  "tac": 101,
11  "rsrp": -85.0,
12  "rsrq": -10.0,
13  "node_id": 5555
14 }

```

GET /cellinfo/signal/ ۲.۵.۲.۳

- عملکرد: بازیابی داده‌ها با فیلترهای اختیاری
- احراز هویت: الزامی
- پارامترها: page_size, page, client_id, end, start, technology

توضیحات کد

۱. خواندن پارامترهای فیلتر از URL

۲. انتخاب مدل و مناسب بر اساس تکنولوژی Serializer

۳. اعمال محدودیت دسترسی:

- کاربران عادی فقط به داده‌های مربوط به خود دسترسی دارند
- ادمین به همه داده‌ها، با امکان فیلتر دسترسی دارند client_id

۴. اعمال فیلتر زمانی (start و end)

۵. صفحه‌بندی داده‌ها (Pagination)

۶. سریالایز کردن داده‌ها و برگرداندن Response

۷. مدیریت خطای در صورت عدم تعیین یا نادرست بودن technology

مثال 2 GET cellinfosignal?technology=LTE&page=1&page_size=2 کاربر

```
1 {
2   "count": 5,
3   "num_pages": 3,
4   "current_page": 1,
5   "results": [
6     {"id": 1, "user": 2, "technology": "LTE", ...},
7     {"id": 2, "user": 2, "technology": "LTE", ...}
8   ]
9 }
```

مثال GET client_id با ادمین

```
1 GET /cellinfo/signal/?technology=LTE&client_id=2&page=1
```

- ادمین می‌تواند داده همه کاربران را ببیند

- با client_id فقط داده کاربر مشخص شده را می‌گیرد

Tests API ۳.۳

برای ثبت و بازیابی تست‌های شبکه کاربران طراحی شده است. تست‌ها شامل:

- PingTest : زمان پاسخ

- DNSTest : زمان پاسخ

- WebResponseTest : زمان پاسخ وب

- HTTPUploadTest / HTTPDownloadTest : سرعت آپلود و دانلود

- SMSTest : زمان ارسال و دریافت پیامک

۱.۳.۳ ویژگی‌ها

- احراز هویت: تمام ویوها IsAuthenticated هستند

- کاربران عادی: فقط داده‌های خودشان را می‌بینند

- ادمین: می‌تواند داده همه کاربران را مشاهده کند و با `client_id` فیلتر کند
- **فیلترها:** `page_size`, `page`, `client_id`, `end`, `start`
- نتایج صفحه‌بندی شده و شامل تعداد کل، تعداد صفحات، صفحه جاری و لیست نتایج: **Pagination**

۲.۳.۳ مدل‌ها

PingTest ۱.۲.۳.۳

- کاربری که تست را ثبت کرده: `user`
- زمان انجام تست: `timestamp`
- زمان پاسخ: `ping_response_time`

DNSTest ۲.۲.۳.۳

- کاربری که تست را ثبت کرده: `user`
- زمان انجام تست: `timestamp`
- زمان پاسخ DNS: `dns_response_time`

WebResponseTest ۳.۲.۳.۳

- کاربری که تست را ثبت کرده: `user`
- زمان انجام تست: `timestamp`
- زمان پاسخ وب: `web_response_time`

HTTPDownloadTest / HTTPUploadTest ۴.۲.۳.۳

- کاربر ثبت‌کننده: `user`
- زمان انجام تست: `timestamp`
- سرعت آپلود یا دانلود به Mbps: `upload_rate` / `download_rate`

SMSTest ۵.۲.۳.۳

- کاربر ثبت‌کننده: `user`
- زمان ارسال پیامک: `timestamp_sent`
- زمان دریافت پیامک: `timestamp_delivery`
- مدت زمان تحویل پیامک: `delivery_duration`
- متن پیامک: `message_content`

Serializers ۳.۳.۳

های Serializer تمام :

PingTestSerializer, DNSTestSerializer, WebResponseTestSerializer, HTTPUploadTestSerializer, HTTPDown- •

loadTestSerializer, SMSTestSerializer

- فیلد user فقط خواندنی است
- اعتبارسنجی داده‌های ورودی برای POST

Views ۴.۳.۳

تمام ویوهای GET از QuerySet مسئول فیلتر کردن، **صفحه‌بندی و سریالایز** استفاده می‌کنند. این Mixin FilteredListMixin است. داده‌ها

POST ۱.۴.۳.۳

- ثبت رکورد جدید تست
- جریان کد:

 ۱. دریافت JSON ورودی
 ۲. سریالایز داده‌ها و اعتبارسنجی
 ۳. ذخیره داده‌ها و اختصاص user به کاربر جاری
 ۴. موفق (201 Created) یا خطأ (400 Bad Request) Response

GET ۲.۴.۳.۳

- بازیابی داده‌ها با فیلتر اختیاری
- جریان کد (داخل `(FilteredListMixin`) دسترسی: URL start ، end ، client_id ، page ، page_size پارامترهای :

 ۱. خواندن پارامترهای `client_id`، `end`، `start`، `page`، `page_size` اعمال محدودیت:
 ۲. کاربران عادی فقط داده‌های خود
 - ادمین همه داده‌ها، امکان فیلتر `client_id`
 - فیلتر زمانی `timestamp` یا `timestamp_sent` برای SMS
 ۳. صفحه‌بندی داده‌ها سریالایز و بازگشت Response
 ۴. سریالایز و بازگشت Response

۵.۳.۳ نمونه‌ها (ورودی و خروجی)

PingTest ۱.۵.۳.۳

/tests/ping POST

```
1 {
2   "timestamp": "2025-08-21T12:00:00Z",
3   "ping_response_time": 25.3
4 }
```

Response Success 201

```
1 {
2   "id": 1,
3   "user": 2,
4   "timestamp": "2025-08-21T12:00:00Z",
5   "ping_response_time": 25.3
6 }
```

GET

/tests/ping/?start=2025-08-21T00:00:00Z&end=2025-08-21T23:59:59Z&page=1&page_size=10

```
1 {
2   "count": 25,
3   "num_pages": 3,
4   "current_page": 1,
5   "results": [
6     {"id": 1, "user": 2, "timestamp": "2025-08-21T12:00:00Z", "ping_response_time": 25.3}
7   ]
8 }
```

DNSTest ۱.۵.۴.۳

/tests/dns/ POST

```
1 {
2   "timestamp": "2025-08-21T12:00:00Z",
3   "dns_response_time": 50.2
4 }
```

Response Success 201

```
1 {
2   "id": 1,
3   "user": 2,
4   "timestamp": "2025-08-21T12:00:00Z",
5   "dns_response_time": 50.2
6 }
```

GET /tests/dns/?start=2025-08-21T00:00:00Z&end=2025-08-21T23:59:59Z

```
1 {
2   "count": 10,
3   "num_pages": 1,
4   "current_page": 1,
5   "results": [
6     {"id": 1, "user": 2, "timestamp": "2025-08-21T12:00:00Z", "dns_response_time": 50.2}
7   ]
8 }
```

WebResponseTest ✓.Δ.¶.¶

/tests/web/ POST

```
1 {
2   "timestamp": "2025-08-21T12:00:00Z",
3   "web_response_time": 120.5
4 }
```

Response Success 201

```
1 {
2   "id": 1,
3   "user": 2,
4   "timestamp": "2025-08-21T12:00:00Z",
5   "web_response_time": 120.5
6 }
```

GET /tests/web/?start=2025-08-21T00:00:00Z&end=2025-08-21T23:59:59Z

```
1 Response Success 200
2 {
```

```

۳   "count": 8,
۴   "num_pages": 1,
۵   "current_page": 1,
۶   "results": [
۷     {"id": 1, "user": 2, "timestamp": "2025-08-21T12:00:00Z", "web_response_time": 120.5}
۸   ]
۹ }

```

API Threshold ۴.۳

برای مدیریت پارامترها و سطح‌های آستانه (Threshold) شبکه طراحی شده است. این آستانه‌ها به کاربر و ادمین اجازه می‌دهند تا کیفیت و کمیت سیگنال شبکه (2G/3G/4G/5G) را بررسی کنند و سطوح رنگبندی شده برای هشدار یا نمایش وضعیت تنظیم کنند.

۱.۴.۳ ویژگی‌ها

- احراز هویت: تمام ویوها `IsAuthenticated` هستند
- کاربران عادی: فقط پارامترهای خودشان را مشاهده و ویرایش می‌کنند
- ادمین: می‌تواند داده همه کاربران را مشاهده کند و با `client_id` فیلتر کند
- POST: ایجاد یا آپدیت پارامترهای Threshold همراه با سطوح آن
- GET: لیست پارامترها و سطوح آنها با امکان فیلتر

۲.۴.۳ مدل‌ها

ThresholdParameter ۱.۲.۴.۳

- `user`: کاربری که این پارامتر را تنظیم کرده
- `name`: نام پارامتر
- `technology`: نوع شبکه (2G/3G/4G/5G)
- `signal_type`: نوع سیگنال (quantity, quality)
- `Relationship`: هر پارامتر می‌تواند چند `ThresholdLevel` داشته باشد

ThresholdLevel ۲.۲.۴.۳

- `parameter`: پارامتر مربوطه
- `level`: شماره سطح (مثلاً ۱، ۲، ۳)

• رنگ مربوط به سطح color

• حداقل مقدار این سطح min_value

• حداکثر مقدار این سطح max_value

Serializers ۳.۴.۳

Input Serializers ۱.۳.۴.۳

• برای هر سطح ThresholdLevelInputSerializer

• شامل نام، نوع سیگنال و لیست سطوح ThresholdParamInputSerializer

• شامل تکنولوژی و پارامترهای Threshold UnifiedThresholdInputSerializer

Output Serializers ۲.۳.۴.۳

• برای نمایش سطح ThresholdLevelSerializer

• نمایش پارامتر و لیست سطوح مرتبط ThresholdParameterSerializer

Views ۴.۴.۳

ThresholdcreateView (POST) ۱.۴.۴.۳

• ثبت یا آپدیت پارامترهای Threshold

• جریان کد:

۱. دریافت JSON ورودی با تکنولوژی و پارامترها

۲. اعتبارسنجی ورودی

۳. بررسی حداقل سطح برای هر پارامتر

۴. اگر پارامتر قبلًا وجود داشته باشد حذف سطوح قدیمی

۵. ایجاد یا آپدیت سطوحها (ThresholdLevel)

۶. موفق (201 Created) یا خطأ (400 Bad Request) Response

POST /thresholds/create/

```
1 {
2   "technology": "4G",
3   "parameters": [
4     {
5       "name": "rsrp",
6       "signal_type": "quantity",
7       "levels": [
```

```

8      {"level": 1, "color": "red", "min": 0, "max": 30},
9      {"level": 2, "color": "yellow", "min": 30, "max": 70},
10     {"level": 3, "color": "green", "min": 70, "max": 100}
11   ]
12 }
13 ]
14 }
```

Response Success 201

```

1 {
2   "message": "Thresholds created/updated successfully."
3 }
```

(سطح کمتر از ۳) Response Fail 400

```

1 {
2   "error": "Parameter 'Signal Strength' must have at least 3 levels."
3 }
```

(GET) ThresholdListView ۲.۴.۴.۳

- نمایش پارامترهای Threshold و سطوح آنها
- فیلترها: client_id, name, technology
- کاربران عادی: فقط پارامترهای خودشان
- ادمین: می‌تواند همه پارامترها را ببیند و با client_id فیلتر کند

GET /thresholds/?technology=4G&name=rsrp

Response Success 200

```

1 [
2   {
3     "user_id": 2,
4     "name": "rsrp",
5     "technology": "4G",
6     "signal_type": "quantity",
7     "levels": [
8       {"level": 1, "color": "red", "min_value": 0, "max_value": 30},
9       {"level": 2, "color": "yellow", "min_value": 30, "max_value": 70},
10      {"level": 3, "color": "green", "min_value": 70, "max_value": 100}
11    ]
12  }
```

